Prof. Martin Hofmann
Dr. Andreas Abel
Dr. Ulrich Schöpp

Ludwig-Maximilians-Universität München
Institut für Informatik
SS 2012

Übungen zur Vorlesung
Formale Sprachen und Komplexität

Blatt 9

Aufgabe 9-1 (2-Keller-Automat, 4 Punkte)
Ein 2-Keller-Automat ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \#, E)$, wobei wie gehabt Z eine Menge von Zuständen, Σ das Eingabealphabet, $\Gamma \supseteq \Sigma$ das Kelleralphabet und $z_0 \in Z$ der Startzustand ist; außerdem ist $E \subseteq Z$ eine Menge von Endzuständen.

Eine Konfiguration ist eine Sequenz $A_n..A_1 z B_1..B_m$, wobei z der aktuelle Zustand des 2-Keller-Automaten und $A_n..A_1$ der Inhalt des ersten Kellers (Zeichen A_1 oben) und $B_1..B_m$ der Inhalt des zweiten Kellers (Zeichen B_1 oben) ist. Das Eingabewort wird auf dem zweiten Keller übergeben und beide Keller sind zu Beginn mit # terminiert. Soll z.B. das Wort $a_1..a_n$ auf Akzeptanz geprüft werden, so ist die initiale Konfiguration $\#z_0 a_1..a_n\#$.

Die (nichtdeterministische) Übergangsfunktion δ hat den Typ $\Gamma \times Z \times \Gamma \to \mathcal{P}_{\mathsf{fin}}(\Gamma^* \times Z \times \Gamma^*)$. Wenn $(C_k..C_1,\ z',\ D_1..D_l) \in \delta(A, z, B)$ (wobei $k, l \geq 0$), so kann der Automat aus der Konfiguration $A_n..A_1 A z B B_1..B_m$ in die Konfiguration $A_n..A_1 C_k..C_1 z' D_1..D_l B_1..B_m$ übergehen. Der Automat akzeptiert, wenn er sich in einem der Endzustände befindet. (Die Keller müssen nicht leer sein.)

Bei der Angabe einer Übergangsfunktion benutzen wir die abkürzende Notation

$$AzB \mapsto C_k..C_1 z' D_1..D_l$$

statt $(C_k..C_1,\ z',\ D_1..D_l) \in \delta(A, z, B)$. (Nur wenn $Z \cap \Gamma = \emptyset$, ansonsten ist die Notation mehrdeutig!)

Zeigen Sie, dass 2-Keller-Automaten Turing-mächtig sind, d.h., dass alle berechenbaren Funktionen als 2-Keller-Automat implementiert werden können. Die Konstruktion des 2-Keller-Automaten beschreiben Sie bitte im Detail, Korrektheitsbeweis ist nicht notwendig.

Aufgabe 9-2 (LOOP- und WHILE-Programme) Die folgenden beiden Programme berechnen nicht das Gleiche.

```
x0 := x1;
LOOP x0 DO
  x0 := x0 + 1
END
```

```
x0 := x1;
WHILE x0 DO
   x0 := x0 + 1
END
```

Geben Sie für beide Programme die berechnete Funktion $f\colon \mathbb{N} \to \mathbb{N}$ an.

Aufgabe 9-3 (Monotonie der Ackermann-Funktion) Sei ack die Ackermann-Péter-Funktion, gegeben durch die Gleichungen:

$$\begin{array}{lcl} \mathsf{ack}(0, \quad y) & = & y+1 \\ \mathsf{ack}(x+1,\, 0) & = & \mathsf{ack}(x, 1) \\ \mathsf{ack}(x+1,\, y+1) & = & \mathsf{ack}(x, \mathsf{ack}(x+1, y)) \end{array}$$

Beweisen Sie für alle $x, y, x', y' \in \mathbb{N}$:

a) $\mathsf{ack}(x, y) > y$.
Hinweis: Beweisen Sie $\forall y.\ \mathsf{ack}(x, y) \geq y + 1$ durch Induktion über x. Im Fall $x + 1$, also um $\mathsf{ack}(x + 1, y) \geq y + 1$ zu zeigen, führen Sie noch eine Nebeninduktion über y.

b) $\mathsf{ack}(x, y + 1) > \mathsf{ack}(x, y)$.
Daraus folgt leicht $\mathsf{ack}(x, y) > \mathsf{ack}(x, y')$ für alle $y > y'$, was sie für die folgenden Beweise benutzen dürfen.

c) $\mathsf{ack}(x + 1, y) \geq \mathsf{ack}(x, y + 1)$.

d) $\mathsf{ack}(x + 1, y) > \mathsf{ack}(x, y)$.

Daraus folgt $\mathsf{ack}(x, y) > \mathsf{ack}(x', y)$ für $x > x'$; damit ist ack strikt monoton in beiden Argumenten.

Aufgabe 9-4 (Beschränkte Ackermann-Funktion ist LOOP-berechenbar, 4 Punkte) Konstruieren Sie eine Folge von LOOP-Programmen $P_0, P_1, \ldots$, so dass P_n die Funktion a_n berechnet, wobei $a_n(m) = \mathsf{ack}(n, m)$.

Sie können ihre Lösungen bis **Montag, den 2.7., um 12:00 Uhr** im Abgabekasten in der Theresienstraße oder über UniWorX abgeben. In UniWorX werden Dateien im `txt`-Format (reiner Text) oder im `pdf`-Format akzeptiert.